Quantifying the implicit process flow abstraction in SBGN-PD diagrams with Bio-PEPA

Laurence Loewe[1] Stuart Moodie[2,1] Jane Hillston[2,1]

[1]Centre for System Biology at Edinburgh
[2]School of Informatics
The University of Edinburgh
Scotland

Laurence.Loewe@ed.ac.uk Stuart.Moodie@ed.ac.uk Jane.Hillston@ed.ac.uk

For a long time biologists have used visual representations of biochemical networks to gain a quick overview of important structural properties. Recently SBGN, the Systems Biology Graphical Notation, has been developed to standardise the way in which such graphical maps are drawn in order to facilitate the exchange of information. Its qualitative Process Diagrams (SBGN-PD) are based on an implicit Process Flow Abstraction (PFA) that can also be used to construct quantitative representations, which can be used for automated analyses of the system. Here we explicitly describe the PFA that underpins SBGN-PD and define attributes for SBGN-PD glyphs that make it possible to capture the quantitative details of a biochemical reaction network. We implemented SBGNtext2BioPEPA, a tool that demonstrates how such quantitative details can be used to automatically generate working Bio-PEPA code from a textual representation of SBGN-PD that we developed. Bio-PEPA is a process algebra that was designed for implementing quantitative models of concurrent biochemical reaction systems. We use this approach to compute the expected delay between input and output using deterministic and stochastic simulations of the MAPK signal transduction cascade. The scheme developed here is general and can be easily adapted to other output formalisms.

1 Introduction

To describe biological pathway information visually has many advantages and Systems Biology Graphical Notation Process Diagrams (SBGN-PD [19]) have been developed for this purpose. Like electronic circuit diagrams, they aim to unambiguously describe the structure of a complex network of interactions using graphical symbols. To achieve this requires both a collection of symbols and rules for their valid combination. SBGN-PD is a visual language with a precise grammar that builds on an underlying abstraction as the basis of its semantics (see p.40 [19]). We call this underlying abstraction for SBGN-PD the "Process Flow Abstraction" (PFA). It describes biological pathways in terms of processes that transform elements of the pathway from one form into another. The usefulness of an SBGN-PD description critically depends on the faithfulness of the underlying PFA and a tight link between the PFA and the glyphs used in diagrams. The graphical nature of SBGN-PD allows only for qualitative descriptions of biological pathways. However, the underlying PFA is more powerful and also forms the basis for quantitative descriptions that could be used for analysis. Such descriptions, however, need to allow the inclusion of the corresponding mathematical details like parameters and equations for computing the frequency with which reactions occur.

Here we aim to make explicit the PFA that already underlies SBGN-PD implicitly. This serves a twofold purpose. First, a better and more intuitive understanding of the underlying abstraction will make it easier for biologists to construct SBGN-PD diagrams. Second, the PFA is easily quantified and making it explicit can facilitate the quantitative description of SBGN-PD diagrams. Such descriptions can then

R.J.Back, I.Petre, E. de Vink (Eds.): Computational Models for Cell Processes (CompMod 2009)
EPTCS 6, 2009, pp. 93–107, doi:10.4204/EPTCS.6.7

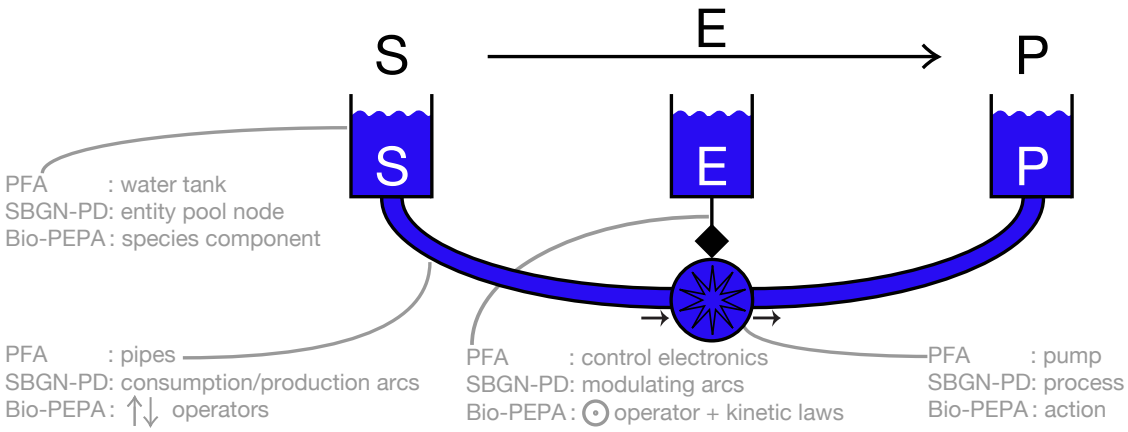


Figure 1: An overview of the process flow abstraction. The chemical reaction at the top is translated into an analogy of water tanks, pipes and pumps that can be used to visualise the process flow abstraction. The various elements are also mapped into SBGN-PD and Bio-PEPA terminology.

be used directly for predicting quantitative properties of the system in simulations. Here we demonstrate how this could work by mapping SBGN-PD to a quantitative analysis system. We use the process algebra Bio-PEPA [5, 1] as an example, but our mapping can be easily applied to other formalisms as well.

The rest of the paper is structured as follows. First we provide an overview of the implicit PFA with the help of an analogy to a system of water tanks, pipes and pumps (Section 2). In Section 3 we explain how this system can be extended in order to capture quantitative details of the PFA. We then show how SBGN-PD glyphs can be mapped to a quantitative analysis framework, using the Bio-PEPA modelling environment [1] as an example (Section 4). In Section 5 we discuss various internal mechanisms and data structures needed for translation into any quantitative analysis framework. As an example of how the translation process works, we apply our new translation tool "SBGNtext2BioPEPA" [20, 21] to a simple model of the MAPK signalling cascade [15], which we automatically translate into Bio-PEPA, where we analyse the stochastic behaviour of the time needed for the cascade to be switched from "off" to "on". We end by reviewing related work and providing some perspectives for further developments.

2 The implicit Process Flow Abstraction of SBGN-PD

The PFA behind SBGN-PD is best introduced in terms of an analogy to a system of many water tanks that are connected by pipes. Each pipe either leads to or comes from a pump whose activity is regulated by dedicated electronics. In the analogy, the water is moved between the various tanks by the pumps. In a biochemical reaction system, this corresponds to the biomass that is transformed from one chemical species into another by chemical reactions. SBGN-PD aims to also allow for descriptions at levels above individual chemical reactions. Therefore the water tanks or chemical species are termed "entities" and the pumps or chemical reactions are termed "processes". For an overview, see Figure 1. We now discuss the correlations between the various elements in the analogy and in SBGN-PD in more detail. In this discussion we occasionally allude to SBGNtext, which is a full textual representation of the semantics of SBGN-PD (developed to facilitate automated translation of SBGN-PD into other formalisms; see [20, 21]). Here are the key elements of the PFA:

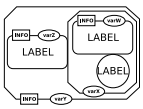
SBGN-PD glyph	EPNType	class type	comment
	Unspecified	material	EPN with unknown specifics
	SimpleChemical	material	EPN
	Macromolecule	material	EPN
	NucleicAcidFeature	material	EPN
	-	material	EPN multimer specified by cardinality
	Complex	container	EPN; arbitrary nesting allowed
	Source	conceptual	external source of molecules
	Sink	conceptual	removal from the system
	PerturbingAgent	conceptual	external influence on a reaction

Table 1: Categories of "water tanks" in the PFA correspond to types of entity pool nodes in SBGN-PD. The complex and the multimers are shown with exemplary auxiliary units that specify cardinality, potential chemical modifications and other information.

Water tanks = entity pool nodes (EPNs). Each water tank stands for a different pool of entities, where the amount of water in a tank represents the biomass that is bound in all entities of that particular type that exist in the system. Typical examples for such pools of identical entities are chemical species like metabolites or proteins. SBGN-PD does not distinguish individual molecules within pools of entities, as long as they are within the same compartment and identical in all other important properties. An overview of all types of EPNs (i.e. categories of water tanks) in SBGN-PD is given in Table 1. To unambiguously identify an entity pool in SBGNtext and in the code produced for quantitative analysis, each entity pool is given a unique `EntityPoolNodeID`. The PFA does not conceptually distinguish between non-composed entities and entities that are complexes of other entities. Despite potentially huge differences in complexity they are all "water tanks" and further quantitative treatment does not treat them differently.

Pipes = consumption and production arcs. Pipes allow the transfer of water from one tank to another. Similarly, to move biomass from one entity pool to another requires the consumption and production of entities as symbolised by the corresponding arcs in SBGN-PD (see Table 3, page 101). These arcs connect exactly one process and one EPN. The thickness of the pipes could be taken to reflect stoichiometry, which is the only explicit quantitative property that is an integral part of SBGN-PD. Production arcs take on a special role in reversible processes by allowing for bidirectional flow.

Pumps = processes. Pumps move water through the pipes from one tank to another. Similarly, processes transform biomass bound in one entity to biomass bound in another entity, i.e., processes transform one entity into another. The speed of the pump in the analogy corresponds to the frequency with which the reaction occurs and determines the amount of water (or biomass) that is transported between tanks (or that is converted from one entity to another, respectively). Processes can belong to different types in SBGN-PD (Table 2) and are unambiguously identified by a unique

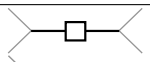
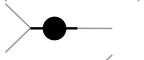
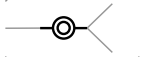
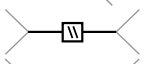
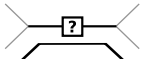
SBGN-PD glyph	ProcessType	meaning
	Process	normal known processes
	Association	special process that builds complexes
	Dissociation	special process that dissolves complexes
	Omitted	several known processes are abstracted
	Uncertain	existence of this process is not clear
	Observable	this process is easily observable

Table 2: Categories of "pumps" in the PFA correspond to types of processes in SBGN-PD.

`ProcessNodeID` in SBGNtext. This allows arcs to clearly define which process they belong to and by finding all its arcs, each process can also identify all EPNs it is connected to.

Reversible processes. SBGN-PD allows for processes to be reversible if they are symmetrically modulated (p.28 [19]). Thus, there may be flows in two directions, however the net flow at any given time will be unidirectional. The PFA does not prescribe how to implement this. For simplicity, our analogy assumes pumps to be unidirectional, like many real-world pumps. Thus bidirectional processes in our analogy are represented as two pumps with corresponding sets of pipes and opposite directions of flow. In our implementation we follow SBGN-PD in separating the left hand side and right hand side of reversible processes for an unambiguous description of reality if all relevant arc glyphs look like production arcs (p.32 [19]). For more details, see [21].

Control electronics for pumps = modulating arcs and logic gates. In the analogy, pumps need to be regulated, especially in complex settings. This is achieved by control electronics. In SBGN-PD, the same is done by various types of modulation arcs, logic arcs and logic gates [19]. They all contribute to determining the frequency of the reaction. Since SBGN-PD does not quantify these interactions, most of our extensions for quantifying SBGN-PD address this aspect. Each arc connects a "water tank" with a given `EntityPoolNodeID` and a "pump" with a given `ProcessNodeID`. Ordinary modulating arcs can be of type `Modulation` (most generic influence on reaction), `Stimulation` (catalysis or positive allosteric regulation), `Catalysis` (special case of stimulation, where activation energy is lowered), `Inhibition` (competitive or allosteric) or `NecessaryStimulation` (process is only possible, if the stimulation is "active", i.e. has surpassed some threshold). The glyphs are shown in Table 3 (page 101), where their mapping to Bio-PEPA is discussed. One might misread SBGN-PD to suggest that Consumption/Production arcs cannot modulate the frequency of a process. However, kinetic laws frequently depend on the concentration of reactants, implying that these arcs can also contribute to the "control electronics" (e.g. report "level of water in tank"). Another part of the "control electronics" are *logical operators*. These simplify modelling, when a biological function can be approximated by a simple on/off logic that can be represented by boolean operators. SBGN-PD supports this simplification by providing the logical operators "AND", "OR" and "NOT", which are connected by "logic arcs" with the rest of the diagram (logic arcs convert to and from the non-boolean world).

Groups of water tanks = compartments, submaps and more. The PFA is complete with all the elements presented above. However, to make SBGN-PD more useful for modelling in a biological context, SBGN-PD has several features that make it easier for biologists to recognise various subsets of entities that are related to each other. For example, entities that belong to the same compartment can be grouped together in the compartment glyph and functionally related entities can be placed on the same submap. In the analogy, this corresponds to grouping related water tanks together. SBGN-PD also supports sophisticated ways for highlighting the inner similarities between entities based on a knowledge of their chemical structure (e.g. modification of a residue, formation of a complex). Stretching the analogy, this corresponds to a way of highlighting some similarities between different water tanks. None of these groupings are important for the PFA in principle or for quantitative analysis, as long as different "water tanks" remain separate.

3 Extensions for quantitative analysis

The process flow abstraction that is implicit in all SBGN process diagrams can be used as a basis to quantify the systems they describe. After we made the PFA explicit above, we now discuss the attributes that need to be added to the various SBGN-PD glyphs in order to allow for automatic translation of SBGN-PD diagrams into quantitative models. These attributes are stored as strings in SBGNtext (our textual representation of SBGN-PD, see [21]) and are attached to the corresponding glyphs by a graphical SBGN-PD editor. They do not require a visual representation that compromises the visual ease-of-use that SBGN-PD aims for. Next we discuss the various attributes that are necessary for the glyphs of SBGN-PD to support quantitative analysis. We do not discuss auxiliary units, submaps, tags and equivalence arcs here, as they do not require extensions for supporting quantitative analysis.

3.1 Quantitative extensions of EntityPoolNodes

For quantitative analysis, each unique EPN requires an `InitialMoleculeCount` to unambiguously define how many entities exist in this pool in the starting state. We followed developments in the SBML standard in using counts of molecules instead of concentrations, since SBGN-PD also allows for multiple compartments, which makes the use of concentrations very cumbersome (see section 4.13.6, p.71f. in [16]). For entities of type `Perturbation`, the `InitialMoleculeCount` is interpreted as the numerical value associated with the perturbation, even though its technical meaning is not a count of molecules. Entities of the type `Source` or `Sink` are both assumed to be effectively infinite, so `InitialMoleculeCount` does not have a meaning for these entities. Beyond a unique `EntityPoolNodeID` and `InitialMoleculeCount`, no other information on entities is required for quantitative analysis.

3.2 Quantitative extensions of Arcs

Arcs link entities and processes by storing their respective IDs and the `ArcType`. The simplest arcs are of type `Consumption` or `Production` and do not require numerical information beyond the stoichiometry that is already defined in SBGN-PD as a property of arcs that can be displayed visually in standard SBGN-PD editors. Logic arcs will be discussed below. All modulating arcs are part of the "control electronics" and affect the frequency with which a process happens. They link to EPNs to inform the process about the presence of enzymes, for example. Modulation is usually governed by parameters or other important quantities for the given process (e.g. Michaelis-Menten-constant).

To make the practical encoding of a model easier, we define process parameters that conceptually belong to a particular modulating entity as a list of `QuantitativeProperties` in the arc pointing to that entity. This is equivalent to seeing the set of parameters of a reaction as something that is specific to the interaction between a particular modulator and the process it modulates. Other approaches are also possible, but lead to less elegant implementations. Storing parameters in equations requires frequent and possibly error-prone changes (e.g. many different Michaelis-Menten equations). One could also argue that the catalytic features are a property of the enzyme and thus make parameters part of EPNs; however this either forces all Michaelis-Menten reactions of an enzyme to happen at the same speed or requires cumbersome naming conventions to manage different affinities for different substrates.

To refer to parameters we specify the `ManualEquationArcID` of an arc and then the name of the parameter that is stored in the list of `QuantitativeProperties` of that arc. This scheme reduces clutter by limiting the scope of the relevant namespace (only few arcs per process exist, so `ManualEquationArcID`s only need to be unique within that immediate neighbourhood). Thus parameter names can be brief, since they only need to be unique within the arc. The `ManualEquationArcID` is specified by the user in the visual SBGN-PD editor and differs from ArcID, a globally unique identifier that is automatically generated by the graphical editor. The `ManualEquationArcID` allows for user-defined generic names that are easy to remember, such as 'Km' and 'vm' for Michaelis-Menten reactions. It should be easily accessible within the graphical editor, just as the parameters that are stored within an arc.

Logical operators and logic arcs. To facilitate the use of logical operators in quantitative analyses one needs to convert the integer molecule counts of the involved EPNs to binary signals amenable to boolean logic. Thus SBGNtext supports "incoming logic arcs" that connect a "source entity" or "source logical result" with a "destination logic operator" and apply an "input threshold" to decide whether the source is above the threshold ("On") or below the threshold ("Off"). To this end, a graphical editor needs to support the "input threshold" as a numerical attribute that the user can enter; all other information recorded in incoming logic arcs is already part of an SBGN diagram. Once all signals are boolean, they can be processed by one or several logical operators, until the result of this operation is given in the form of either 0 ("Off") or 1 ("On"). This result then needs to be converted back to an integer or float value that can be further processed to compute process frequencies. Thus a graphical editor needs to support corresponding attributes for defining a low and a high output level.

3.3 Quantitative extensions of ProcessNodes

For quantitative analyses, a ProcessNode must have a unique name and an equation that computes the propensity, which is proportional to the probability that this process occurs next, based on the current global state of the model. Since the `ProcessType` is not required for quantitative analyses, it does not matter whether a process is an ordinary `Process`, an `Uncertain` process or an `Observable` process, for example. For all these ProcessNodes, graphical editors need to support attributes for the manual specification of a `ProcessNodeID`, and a `PropensityFunction`. These attributes are then stored in SBGNtext. If support for bidirectional processes is desired, then graphical editors need to facilitate entering a propensity function for the backward process as well. Propensity functions compute the propensity of a unidirectional process to be the next event in the model and can be used directly by simulation algorithms and solvers [12].

To instantiate the propensity function, a translator needs to replace all aliases by their true identity. We use the following syntax for a parameter alias that is substituted by the actual numeric value (or a globally defined parameter) from the corresponding arc:

<par: ManualEquationArcID.QuantitativePropertyName>

While translating to Bio-PEPA this would be simply substituted with a corresponding parameter name. The parameter is then defined elsewhere in the Bio-PEPA code to have the numerical value stored in the corresponding property of the arc. To allow the numerical analysis tool to access an EPN count at runtime we replace the following entity alias by the EntityPoolNodeID that the corresponding arc links to:

<ent: ManualEquationArcID >

This is shorter than the EntityPoolNodeID and allows the reuse of propensity functions if kinetic laws are identical and the manual IDs follow the same pattern. It is desirable that there is no need to specify the EntityPoolNodeID. It is fairly long and generated automatically to reflect various properties that make it unique. It would be cumbersome to refer to in the equation and it would require a mechanism to access the automatically generated EntityPoolNodeID before a SBGNtext file is generated. Also any changes to an entity that would affect its EntityPoolNodeID would then also require a change in all corresponding propensity functions, a potentially error-prone process. The same substitution mechanism can be used to provide access to properties of compartments (see [21]).

In addition to these aliases, functions use the typical standard arithmetic rules and operators that are merely handed through to the analysis tool.

4 Mapping SBGN-PD elements to Bio-PEPA

In this section we explain how to use the semantics of SBGN-PD to map a SBGN-PD model to a formalism for the quantitative analysis of biochemical systems. We are using Bio-PEPA as an example, but our approach is general and can be applied to many other formalisms.

4.1 The Bio-PEPA language

Bio-PEPA is a stochastic process algebra which models biochemical pathways as interactions of distinct entities representing reactions of chemical species [5, 1]. A process algebra model captures the behaviour of a system as the actions and interactions between a number of entities, where the latter are often termed "processes", "agents" or "components". In PEPA and Bio-PEPA these are built up from simple *sequential components* [5, 14, 3]. Different process algebras support different modelling styles for biochemical systems [3]. Stochastic process algebras, such as PEPA [14] or the stochastic π-calculus [24], associate a random variable with each action to represent the mean of its exponentially distributed waiting time. In the stochastic π-calculus, interactions are strictly binary whereas in Bio-PEPA the more general multiway synchronisation is supported. The syntax of Bio-PEPA is defined as [5] :

$$S ::= (\alpha,\kappa)\ \mathsf{op}\ S \mid S + S \mid C \qquad\qquad P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(x)$$

where S is a sequential *species component* that represents a chemical species (termed "process" in some other process algebras and "EntityPoolNode" in SBGN-PD), C is a constant pointing to an S, P is a *model component* that describes the set $\mathcal{L}$ of possible interactions between species components (these "interactions" or "actions" correspond to "processes" in SBGN-PD and can represent chemical reactions). A count of molecules or a concentration of S is given by $x \in \mathbb{R}_0^+$. In the prefix term "$(\alpha,\kappa)\ \mathsf{op}\ S$", κ is the *stoichiometry coefficient* and the operator op indicates the role of the species in the reaction α.

Specifically, op $= \downarrow$ denotes a *reactant*, $\uparrow$ a *product*, $\oplus$ an *activator*, $\ominus$ an *inhibitor* and $\odot$ a generic *modifier*, which indicates more complex roles than $\oplus$ or $\ominus$. The operator "+" expresses a choice between possible actions. Finally, the process $P \underset{\mathcal{L}}{\bowtie} Q$ denotes the synchronisation between components: the set $\mathcal{L}$ determines those activities on which the operands are forced to synchronise. When $\mathcal{L}$ is the set of common actions, we use the shorthand notation $P \bowtie Q$.

A Bio-PEPA system $\mathcal{P}$ is defined as a 6-tuple $\langle \overset{*}{\mathcal{V}}, \mathcal{N}, \mathcal{K}, \mathcal{F}_R, Comp, P \rangle$, where: $\mathcal{V}$ is the set of compartments, $\mathcal{N}$ is the set of quantities describing each species (includes the initial concentration), $\mathcal{K}$ is the set of all parameters referenced elsewhere, $\mathcal{F}_R$ is the set of functional rates that define all required kinetic laws, *Comp* is the set of species components S that highlight the reactions an entity can take part in and P is the system model component. Bio-PEPA models (i) represent reversible reactions as pairs of irreversible forward and backward reactions, (ii) treat the same species in different states or compartments as different species represented by distinct Bio-PEPA components and (iii) assume static compartments. See [5] for more details.

A variety of analysis techniques can be applied to a single Bio-PEPA model, facilitating the easy validation of analysis results when the analyses address the same issues [2] and enhancing insight when the analyses are complementary [4]. Currently supported analysis techniques include stochastic simulation at the molecular level, ordinary differential equations, probabilistic model checking and numerical analysis of continuous time Markov chains [5, 1, 9].

4.2 SBGN-PD mapping

Here we map the core elements of SBGN-PD to Bio-PEPA (see [20] for an implementation).

Entity Pool Nodes Due to the rich encoding of information in the `EntityPoolNodeID`, Bio-PEPA can treat each distinct `EntityPoolNodeID` as a distinct species component. This removes the need to explicitly consider any other aspects such as entity type, modifications, complex structures and compartments, as all such information is implicitly passed on to Bio-PEPA by using the `EntityPoolNodeID` as the name for the corresponding species component. To define the set $\mathcal{N}$ of a Bio-PEPA system requires the attribute `InitialMoleculeCount` for each EPN (see Section 3).

Processes All SBGN-PD `ProcessTypes` are simply represented as reactions in Bio-PEPA. Compiling the corresponding set $\mathcal{F}_R$ relies on the attribute `PropensityFunction` and a substitution mechanism that makes it easy to define these functions manually. To help humans understand references to processes in the sets $\mathcal{F}_R$ and *Comp* requires recognizable names for SBGN-PD `ProcessNodeID`s that map directly to their identifiers in Bio-PEPA. Thus graphical editors need to allow for manual `ProcessNodeID`s.

Reversible processes. The translator supports reversible SBGN-PD processes by dividing them into two unidirectional processes for Bio-PEPA. The translator reuses the manually assigned `ProcessNodeID` and augments it by "_F" for forward reactions and "_B" for backward reactions. These two unidirectional processes are then treated independently. When compiling the species components in Bio-PEPA, every time a `LeftHandSide` arc is found, the translator assumes that the corresponding forward and backward processes have been defined and will augment the process name by "_F" for forward reactions and "_B" for backward reactions, while adding the corresponding Bio-PEPA operator for reactant and product. `RightHandSide` arcs are handled in the same way. Thus the production arc glyph in SBGN-PD has three distinct meanings as shown in Table 3.

Arcs. The arcs in SBGN-PD define which entities interact in which processes. Thus arcs play a pivotal role in defining the species components in Bio-PEPA. Since arcs can store kinetic parameters, they are also important for defining parameters in Bio-PEPA. As kinetic law definitions in Bio-PEPA frequently refer to such parameters, we use the ArcID that is automatically generated by the graphical

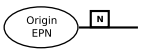
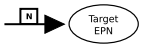
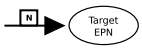
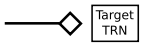
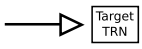
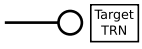
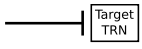
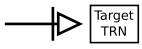
SBGN-PD glyph	ArcType	Bio-PEPA symbol	Bio-PEPA code
	Consumption	↓	<<
	Production	↑	>>
	LeftHandSide	↓ and ↑	<< and >>
	RightHandSide	↑ and ↓	>> and <<
	Modulation	⊙	(.)
	Stimulation	⊕	(+)
	Catalysis	⊕	(+)
	Inhibition	⊖	(-)
	NecessaryStimulation	⊙	(.)

Table 3: "Water pipes and control electronics": Mapping arcs between entities and processes in SBGN-PD to operators in Bio-PEPA species components. The "symbols" are the formal syntax of Bio-PEPA, while the "code" gives the concrete syntax used in the Bio-PEPA Eclipse Plug-in [1].

editor to substitute the local manual arc references in propensity functions by globally unique parameters names (see Section 3). The type of an arc indicates both the role of the connected entity in the process (consumed reactant, product or modifier of rate) and the chemical nature of the reaction (catalysis, stimulation, inhibition, necessary stimulation or the most generic modification). Thus the type of an arc can be mapped directly to the operator "op" described in the Bio-PEPA syntax shown in Table 3.

Logical operators Logical operators require the conversion of integer molecule counts of the relevant EPNs to binary signals and after some boolean logic processing back to low and high integer values. As evident from the implementation scheme above, the use of all quantitative properties culminates in the correct formulation of the corresponding propensity functions that determine the probability that the corresponding process will be the next to occur. Thus an implementation of logical operators requires that their results be included in the corresponding propensity functions. The current scheme of implementing propensity functions relies heavily on substituting the various components into the final equation, so that Bio-PEPA will ultimately only see one formula per propensity function. In this context the implementation of logical operators requires the insertion of a formula in the propensity function that computes the result of the boolean operations from their integer input. An arbitrarily complex logic operator network can be constructed from the following basic building blocks:

- *Convert from integers or double floats to boolean values*. This is best done by a specially defined mathematical function that takes an integer or float signal and compares it to a specified threshold, giving back either 0 (signal < threshold) or 1 (signal ≥ threshold). The definition of such a function is not complicated and is implemented in the current Bio-PEPA Eclipse Plug-in (starting with version 0.1).
- AND operator. Multiply all boolean inputs to arrive at the output.
- NOT operator. The arithmetic expression (1- *input*) computes the *output*
- OR operator. Sum all inputs (0/1) and test if it is greater than 1 using the threshold function.

Of these, only the threshold function is not widely available, even if it is easy to implement.

5 Converter implementation and internal representation

We chose Java as implementation language for the converter described above, due to the good portability of the resulting binaries and the use of Java in the Bio-PEPA Eclipse Plug-in [1]. We defined a grammar for SBGNtext in the Extended Backus-Naur-Form (EBNF) as supported by ANTLR [22], which automatically compiles the Java sources for the corresponding parser that stores all important parsing results in a number of coherently organised internal TreeMaps. To compile a working Bio-PEPA model three main loops over these TreeMaps are necessary: over all entities, over all processes and over all parameters. To illustrate the translation we refer to "`code`" examples from Figure 2.

The **loop over all entities** (e.g. "`m_MAPKK_PP`") compiles the species components as well as the model description required by Bio-PEPA. The latter is a list of all participating `EntityPoolNodeID`s combined by the cooperation operator "$< * >$" that automatically synchronises all common actions. This simplification depends on all processes in SBGN-PD having unique names and fixed lists of reactants with no mutually exclusive alternatives in them. The first condition can be enforced by the tools that produce the code, the second is ensured by the reaction-style of describing processes in SBGN-PD.

For each species component a loop over all arcs finds the arcs that are connected to it (e.g. "`st27`") and that store all relevant `ProcessNodeID`s (e.g. "`K_P_act`"). The same loop determines the respective role of the component (as reflected by the choice of the Bio-PEPA operator in Table 3; e.g. "`(+)`").

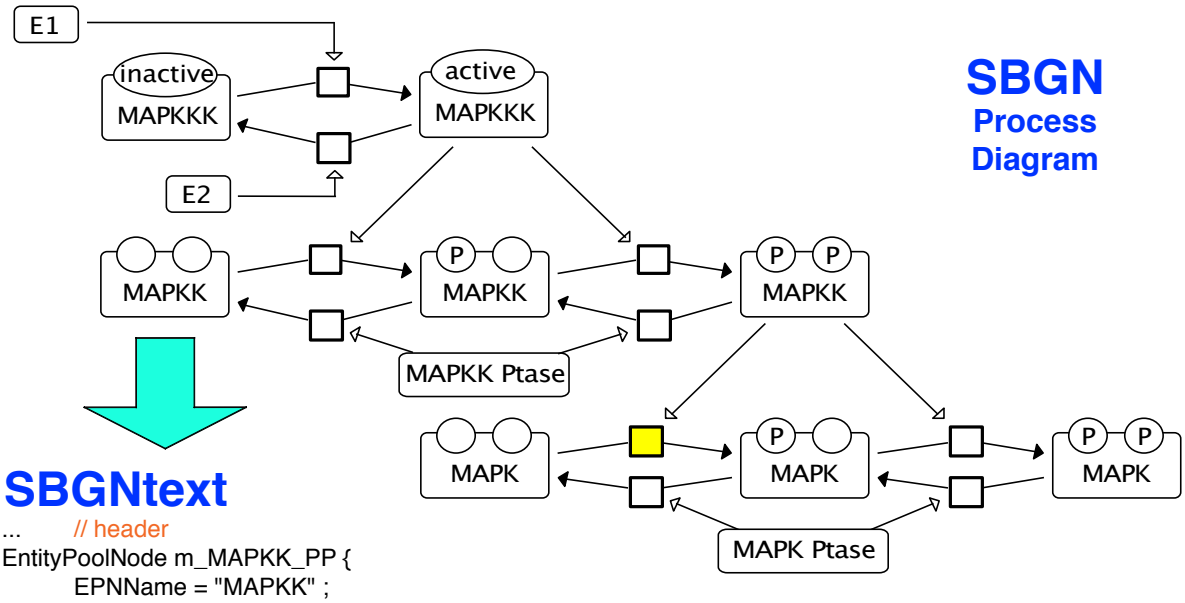
The **loop over all processes** (e.g. "`K_P_act`") compiles the kinetic laws by substituting aliases (e.g. "`<par: enz.kcat>`" or "`<ent: enz>`") for parameters ("`st27_kcat`") and EPNs ("`m_MAPKK_PP`") in the propensity functions specified in the graphical editor. Each function is handled separately by a dedicated function parser that queries the TreeMaps generated when parsing the SBGNtext file.

The **loop over all quantitative properties** of the model defines the parameters in Bio-PEPA (e.g. "`st27_kcat`"). It is possible to avoid this step by inserting the direct numerical values into the equations processed in the second loop. However, this substantially reduces the readability of equations in the Bio-PEPA code and makes it difficult for third party tools to assist in the automated generation of parameter combinations. Thus we defined a scheme that automatically generates parameter names to maximise the readability of equations (combine ArcID "`st27`" and name of the quantitative property "`kcat`").

To facilitate walking over the various collections specified above, the TreeMaps are organised in four sets, one for entities, one for processes, one for arcs and one for quantitative properties. Each of these sets is characterised by a common key to all maps within the set. This facilitates the retrieval of related parse products for the same key from a different map. Since all this information is accessible from Java code, it is easily conceivable to use the sources produced in this work for reading SBGNtext files in a wide variety of contexts. The system is easy to deploy, since it involves few files and the highly portable ANTLR runtime library. Our converter is called SBGNtext2BioPEPA and sources are available [20].

6 Example: Stochasticity in the MAPK cascade

Here we illustrate our translation by applying SBGNtext2BioPEPA [20] to a real-world example. We implemented a simple model of the Mitogen-Activated Protein Kinase (MAPK) signal transduction cascade [18, 15] in order to investigate the time needed for the cascade to switch from "off" to "on". The general pattern of the MAPK cascade is well conserved across many biological taxa and triggers a highly varied range of molecular responses [15]. Such a broad conservation suggests not only important functionality (maintained by purifying selection), but also an astonishing flexibility in how a MAPK cascade might be implemented (it has to work in a wide range of contexts). For example, MAPK cascades operate in



SBGNtext

```
...      // header
EntityPoolNode m_MAPKK_PP {
        EPNName = "MAPKK" ;
        EPNType =  Macromolecule  ;
        EPNState = { ( Label = "P" ) ( Label = "P" ) } ;
        InitialMoleculeCount =  0 ;
} ...    // 11 more EPNs
ProcessNode K_P_act {
        ProcessType = Process;
        PropensityFunction =
          "<par: enz.kcat>*<ent: enz>*<ent: subs>
        / (<par: enz.Km_CountCell>*<ent: subs>)" ;
} ...    // 9 more ProcessNodes
Arc st27 {
        ManualEquationArcID = enz ;
        ArcType = Stimulation ;
        Entity = m_MAPKK_PP  ;
        Process = K_P_act  ;
        Stoichiometry = 1 ;
        QuantitativeProperties = {
                ( Km_CountCell =   300.0  )
                ( kcat =  10.0  )
        } ;
} ...    // 29 more Arcs
...      // ending
```

Bio-PEPA

```
st27_Km_CountCell = 300;
st27_kcat = 10.0;
...      // 18 more parameters

kineticLawOf  K_P_act :
        st27_kcat  *
        m_MAPKK_PP  *  m_MAPK_P   /
        ( st27_Km_CountCell  *  m_MAPK_P )
;  ...   // 9 more kinetic laws

m_MAPKK_PP =
        ( K_act          , 1 ) (+) m_MAPKK_PP  +
        ( K_P_act        , 1 ) (+) m_MAPKK_PP  +
        ( KK_P_act       , 1 ) >> m_MAPKK_PP  +
        ( KK_PP_deact  , 1 ) << m_MAPKK_PP
;  ...   // 11 more species components

// model component with initial counts per cell
m_MAPKK_PP [0] <*> m_MAPK_P [0] <*> ...
   // 9 more species components [initial count]
```

Figure 2: A possible SBGN-PD representation of a MAPK cascade with fragments from the corresponding SBGNtext code and the automatically generated Bio-PEPA code. Most biologists would find it far more natural to draw the diagram at the top than to write the Bio-PEPA code. The code excerpts focus on EPN MAPK-PP and the reaction it catalyses (`'K_P_act'`, yellow node). All parameters are scaled to represent counts of molecules per cell (including the Michaelis-Menten constants). For the full code and newer versions see [20]. Biologically, the input signal is a change in `E1`, the output signal a change in `MAPK_PP`. The back reactions on each of the three levels can be thought of as a "conveyer belt" that constantly turns active molecules (on the right) into passive ones (on the left). `E1` must overcome the conveyer belt on the first level for sequentially overcoming the other conveyer belts too. In many such systems either active or inactive molecules are essentially absent.

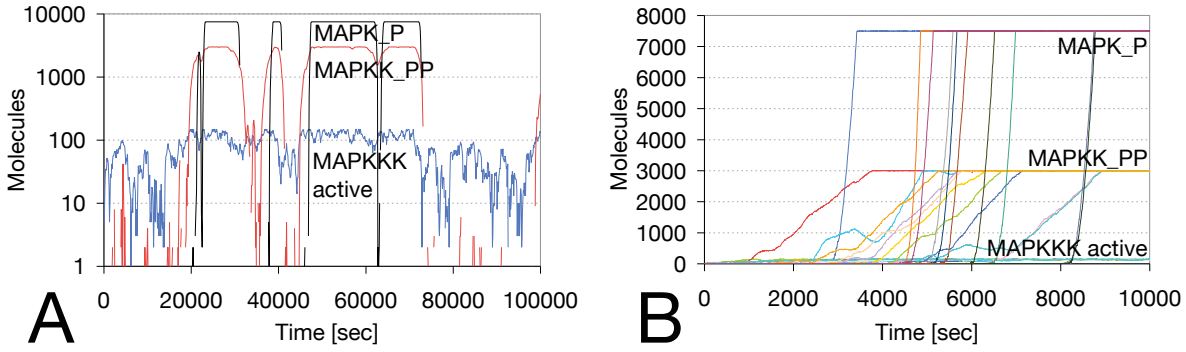


Figure 3: Noise in the MAPK cascade. (A) One stochastic simulation trace, where E1 = E2 = 20 makes the cascade switch between the "on" and "off"state, as activating and deactivating reactions at the first level are equally strong. (B) Ten stochastic simulation traces, where E1 = 21, E2 = 20. If activating reactions are only slightly stronger than deactivating reactions at the first level of the cascade, the cascade can be expected to switch "on". However the time until the "on" state is reached is subject to considerable stochasticity, which is mostly determined by stochasticity during the initial stages of the switching processes.

oocytes of the frog *Xenopus* and in yeast cells, despite their 5 million fold differences in size [15, 27] and substantial differences in kinetic constants [15, 17].

Figure 2 shows a SBGN-PD diagram of the basic structure of a MAPK cascade. It accepts "input" in the form of a change in the concentration of the enzyme E1 and then propagates the signal by changing the concentrations of the various intermediate substrates and enzymes until the "output" enzyme MAPK_PP is affected. If the input is off (low E1), the output is off too (no MAPK_PP). If the input is on (E1 above threshold), output is reliably on as well (high MAPK_PP). Important properties for signalling cascades include the reliability of transmitting a signal and the speed with which this happens. Building on the model in [15], a recent study explored the expected percentage of active output and the time until all output is activated [18].

Here we automatically translate our SBGNtext model of the MAPK cascade via SBGNtext2BioPEPA into a Bio-PEPA model (see [20] for code), which we analyse with the Bio-PEPA Eclipse Plug-in [1]. We report the results of stochastic simulations using the Gibson-Bruck algorithm [11, 12]. Each reaction is assumed to follow Michaelis-Menten kinetics, resulting in the following propensity function:

$$\frac{k_{cat}[count/sec] * Enzyme[count] * Substrate[count]}{K_M[count] + Substrate[count]}$$

where [count] indicates the absolute number of molecules of this entity within the cell, k_{cat} denotes the number of substrate molecules that can be processed by one enzyme at maximal speed, K_M denotes the Michaelis-Menten constant that is an inverse measure of the affinity between enzyme and substrate. We assume $K_M = 300$ and $k_{cat} = 10$ for all reactions (see [15, 17] for other possible values). All our simulations start with molecule counts of E2 = 20, MAPKKK = 150, MAPKK = 3000, MAPKK_Ptase = 100, MAPK = 7500,MAPK_Ptase = 2000 and 0 for all other entities, reflecting the equilibrium "off" state. These molecule counts roughly reflect MAPK cascades in yeast, and are about 6 orders of magnitude below corresponding counts in *Xenopus* oocytes [15]. We tested various E1 input signals and found the following results that cannot be obtained without translating SBGN-PD into a quantitative formalism.

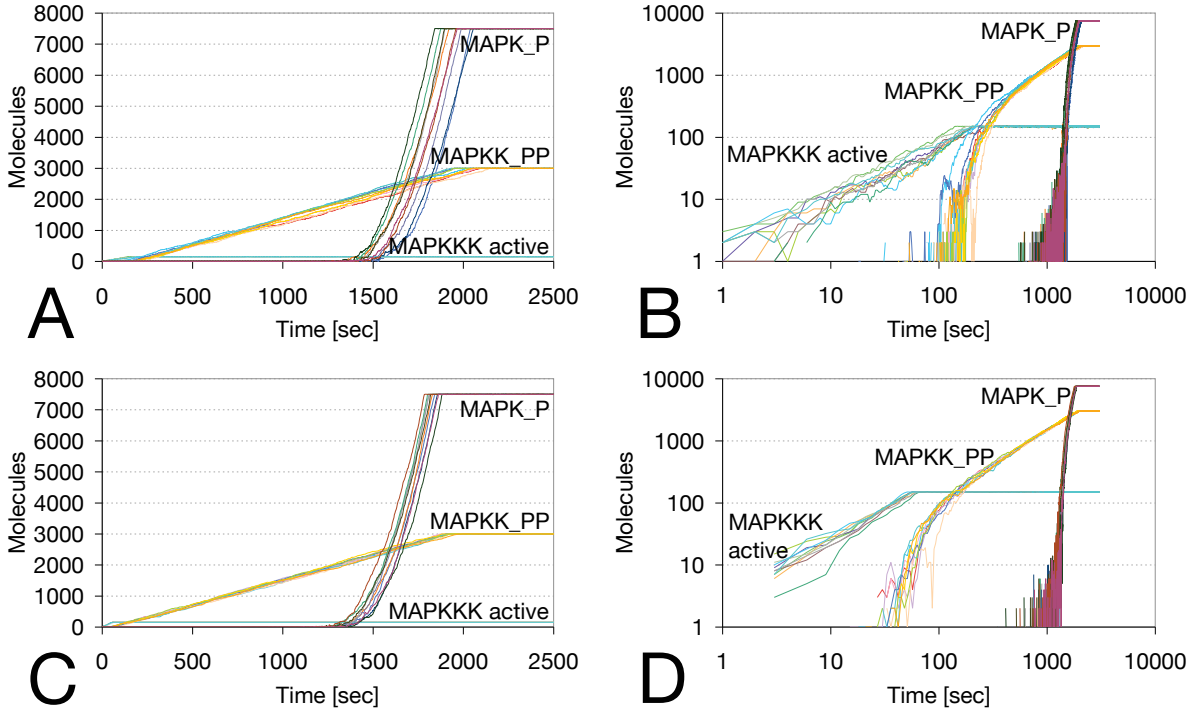


Figure 4: Increasing the number of E1 molecules from 40 (upper part, A, B) to 100 (lower part, C, D) reduces the stochasticity of the time to the completion of the switch (see linear plots on the left, A, C). The stochasticity in the initial stages is considerable in both cases for this system (see log plots on the right, B, D). Ten stochastic simulation traces are shown. E2 = 20; for other parameters, see text.

1. The cascade remains "off", if E1<E2 (not shown).
2. The cascade switches beween "on" and "off", if E1 = E2 as shown in Figure 3A.
3. The cascade switches reliably to "on" if E1>E2, but at E1 = 21 there is considerable noise in the "signalling time" (from switching E1 "on" until all output MAPK_PP is switched "on" as well; see Figure 3B).
4. Further increasing E1 substantially reduces the variability in signalling times and slightly improves speed (Figure 4).
5. Additional tests have shown that the overall signalling speed is very strongly influenced by k_{cat} (not shown).

7 Related work

There are various tools that map visual diagrams to quantitative modelling environments (e.g. SPiM [23], BlenX [7], kappa [6], Snoopy [13], EPN-PEPA [26], JDesigner [25]). However the corresponding graphical notations are not as rich as SBGN-PD and are thus not easily applied to the wide range of

scenarios that SBGN-PD was designed for. Since SBGN-PD is emerging as a new standard, it is clearly desirable to translate from SBGN-PD to a quantitative environment.

Since the first draft of SBGN-PD has been published in August 2008, a number of tools are being developed to support it. The graphical editor CellDesigner [10] supports a subset of SBGN-PD and can translate it into SBML which is supported by many quantitative analysis tools. However the process of adding quantitative information involves cumbersome manual interventions. This motivated work for SBMLsqueezer [8], a CellDesigner plug-in that supports the automatic construction of generalised mass action kinetics equations. While the automated suggestions for the kinetic laws from SBMLsqueezer might be of interest for some problems, the generated reactions contain many parameters that are extraordinarily difficult to estimate. Thus it is preferable to also allow the user to enter arbitrary kinetic laws that may have to be hand-crafted, but whose equations are simpler and require fewer parameter estimates. In SBGNtext2BioPEPA this is combined with mechanisms to reuse the code for such kinetic laws, greatly reducing practical difficulties and the potential for errors.

8 Conclusion and Perspectives

We have explicitly described the process flow abstraction that implicitly underlies SBGN-PD. We have used this abstraction to design a mechanism for translating SBGN-PD into code that can be used for quantitative analysis, using Bio-PEPA as an example. In order to do this we build on SBGNtext, a textual representation of SBGN-PD that we created [20, 21] and that focusses on the key functional SBGN-PD content, avoiding the clutter that comes from storing graphical details. We have developed our translator SBGNtext2BioPEPA in Java to facilitate its integration in the Systems Biology Software Infrastructure that is currently under development at the Centre for Systems Biology at Edinburgh (http://csbe.bio.ed.ac.uk/). SBGNtext2BioPEPA contains a parser for our SBGNtext format based on a formal ANTLR EBNF grammar and is freely available [20]. Building on the process flow abstraction and the internal representation of entities, processes, arcs and parameters in our code can make it easy to implement translations of SBGNtext to other modelling platforms. Since biologists are much more comfortable with drawing visual diagrams than writing code, support for translating SBGN-PD into quantitative analysis systems can play a key role in facilitating quantitative modelling.

Acknowledgements. We thank Stephen Gilmore for comments that improved this manuscript. The Centre for Systems Biology Edinburgh is a Centre for Integrative Systems Biology (CISB) funded by BBSRC and EPSRC, reference BB/D019621/1.

References

[1] Bio-PEPA homepage `http://www.biopepa.org/`. To install the Bio-PEPA Eclipse Plug-in by Adam Duguid follow the links from `http://homepages.inf.ed.ac.uk/jeh/Bio-PEPA/Tools.html` (2009)

[2] Calder M. , Duguid A., Gilmore S. and Hillston J.: Stronger computational modelling of signalling pathways using both continuous and discrete-space methods. Proc. of *CMSB'06*, LNCS vol. 4210, pp. 63–77 (2006)

[3] Calder, M. and Hillston, J.: Process algebra modelling styles for biomolecular processes. *Transactions on Computational Systems Biology*, in press, (2009)

[4] Ciocchetta F., Gilmore S., Guerriero M.-L. and Hillston J.: Stochastic Simulation and Probabilistic Model-Checking for the Analysis of Biochemical Systems. To appear in ENTCS (Proc. of PASM 2008).

[5] Ciocchetta F. and Hillston J.: A framework for the modelling and analysis of biological systems. Theoretical Computer Science, doi:10.1016/j.tcs.2009.02.037 (2009)

[6] Danos V., Feret J., Fontana W., Harmer R., Krivine J.: Rule-based modelling of cellular signalling. Lecture Notes in Computer Science, vol. 4703, pp.17-41, (2007).

[7] Dematté L., Priami C., Romanel A.: The BlenX Language: A Tutorial. SFM 2008, LNCS 5016:313-365, Springer (2008)

[8] Draerger A., Hassis N., Supper J., Schröder A., and Zell A.: SBMLsqueezer: A CellDesigner plug-in to generate kinetic rate equations for biochemical networks. BMC Systems Biology, 2:39 (2008)

[9] Duguid A., Gilmore S., Guerriero M.L., Hillston J. and Loewe L.: Design and development of software tools for Bio-PEPA. Proceedings of the 2009 Winter Simulation Conference (WSC'09), eds. M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, Austin, Texas, to appear (2009).

[10] Funahashi, A.; Matsuoka, Y.; Jouraku, A.; Morohashi, M.; Kikuchi, N.; Kitano, H.: CellDesigner 3.5: A Versatile Modeling Tool for Biochemical Networks, Proceedings of the IEEE vol. 96, issue 8, pp. 1254-1265, http://www.celldesigner.org/ (2008)

[11] Gibson M.A., Bruck J.: Efficient exact stochastic simulation of chemical systems with many species and many channels. Journal of Physical Chemistry A, 104:1876-1889 (2000)

[12] Gillespie, D. T.: Stochastic simulation of chemical kinetics. Annu. Rev. Phys. Chem. 58:35-55 (2007)

[13] Heiner, M., Richter R., Schwarick M., Rohr C.: Snoopy - A tool to design and execute graph-based formalisms. Petri Net Newsletter 74 (April) ISSN 0931-1084, pp. 8-22, http://www-dssz.informatik.tu-cottbus.de/software/snoopy.html (2008)

[14] Hillston, J.: *A Compositional Approach to Performance Modelling*, Cambridge University Press (1996)

[15] Huang C.Y., Ferrell J.E., Jr.: Ultrasensitivity in the mitogen-activated protein kinase cascade. Proc Natl Acad Sci USA, 93:10078-10083, (1996)

[16] Hucka M., et al.: Systems Biology Markup Language (SBML) Level 2 Version 4 Release 1. Nature Precedings http://dx.doi.org/10.1038/npre.2008.2715.1 and http://sbml.org/ (2008)

[17] Kholodenko, B.N.: Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascades. Eur J Biochem vol. 267 (6) pp. 1583-8 (2000)

[18] Kwiatkowska M., Norman G. and Parker D.: Using probabilistic model checking in systems biology. ACM SIGMETRICS Performance Evaluation Review vol. 35 (4) 14-21 (2008)

[19] Le Novère, N. et al.: Systems Biology Graphical Notation: Process Diagram Level 1. Nature Preceedings http://hdl.handle.net/10101/npre.2008.2320.1 (2008)

[20] Loewe L.: SBGNtext2BioPEPA. http://csbe.bio.ed.ac.uk/SBGNtext2BioPEPA/index.php (2009)

[21] Loewe L., Moodie S., Hillston J.: Technical Report: Defining a textual representation for SBGN Process Diagrams and translating it to Bio-PEPA for quantitative analysis of the MAPK signal transduction cascade. Technical Report of the School of Informatics, University of Edinburgh
http://csbe.bio.ed.ac.uk/SBGNtext2BioPEPA/index.php (2009)

[22] Parr, T.: The Definitive ANTLR Reference: Building Domain-Specific Languages. The Pragmatic Bookshelf, Raleigh, NC. http://www.antlr.org/ (2007)

[23] Phillips A.: A Visual Process Calculus for Biology. Jones and Bartlett Publishers, to appear (2009) http://research.microsoft.com/en-us/projects/spim/

[24] Priami C.: Stochastic π-calculus. *The Computer Journal*, 38(6), pp. 578–589, (1995)

[25] Sauro H.M., Hucka M., Finney A., Wellock C., Bolouri H., Doyle J. and Kitano H.: Next generation simulation tools: the Systems Biology Workbench and BioSPICE integration. OMICS. 2003 7(4):355-72, (2003). For the graphical front end JDesigner see: http://www.sys-bio.org/software/jdesigner.htm

[26] Shukla, A.: Mapping the Edinburgh Pathway Notation to the Performance Evaluation Process Algebra. MSc. Thesis, University of Trento (2007)

[27] Wallace R.A., Misulovin Z.: Long-term growth and differentiation of Xenopus oocytes in a defined medium. P Natl Acad Sci USA vol. 75 (11) pp. 5534-8 (1978)